Introduction to Relational Databases

Introduction

- Database – collection of persistent data
- Database Management System (DBMS) – software system that supports creation, population, and querying of a database

Relational Database

- Relational Database Management System (RDBMS)
  - Consists of a number of *tables* and single *schema* (definition of tables and attributes)
  - Students (sid, name, login, age, gpa)

    Students identifies the table

    sid, name, login, age, gpa identify attributes

    sid is primary key

An Example Table

- Students (*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real)

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Another example: Courses

- Courses (cid, instructor, quarter, dept)

cid	instructor	quarter	dept
Carnatic101	Jane	Fall 06	Music
Reggae203	Bob	Summer 06	Music
Topology101	Mary	Spring 06	Math
History105	Alice	Fall 06	History

Keys

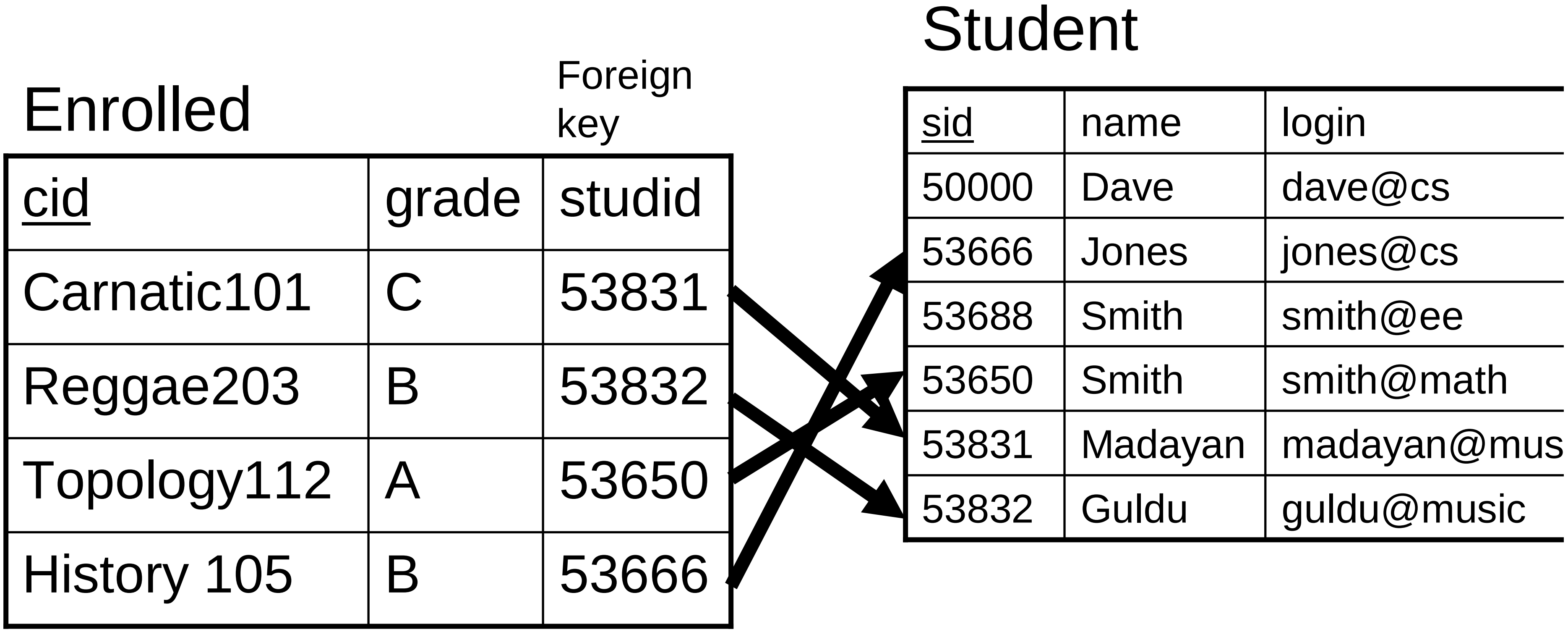
- Primary key – minimal subset of fields that is unique identifier for a tuple
  - sid is primary key for Students
  - cid is primary key for Courses
- Foreign key –connections between tables
  - Courses (cid, instructor, quarter, dept)
  - Students (sid, name, login, age, gpa)
  - How do we express which students take each course?

Many to many relationships

- In general, need a new table

  Enrolled(cid, grade, studid)

  Studid is *foreign key* that references sid in Student table



Enrolled

Foreign key

cid	grade	studid
Carnatic101	C	53831
Reggae203	B	53832
Topology112	A	53650
History 105	B	53666

Student

sid	name	login
50000	Dave	dave@cs
53666	Jones	jones@cs
53688	Smith	smith@ee
53650	Smith	smith@math
53831	Madayan	madayan@mus
53832	Guldu	guldu@music

Relational Algebra

- Collection of operators for specifying queries
- Query describes step-by-step procedure for computing answer (i.e., *operational*)
- Each operator accepts one or two relations as input and returns a relation as output
- Relational algebra expression composed of multiple operators

Basic operators

- Selection – return *rows* that meet some condition
- Projection – return *column* values
- Union
- Cross product
- Difference
- Other operators can be defined in terms of basic operators

Example Schema (simplified)

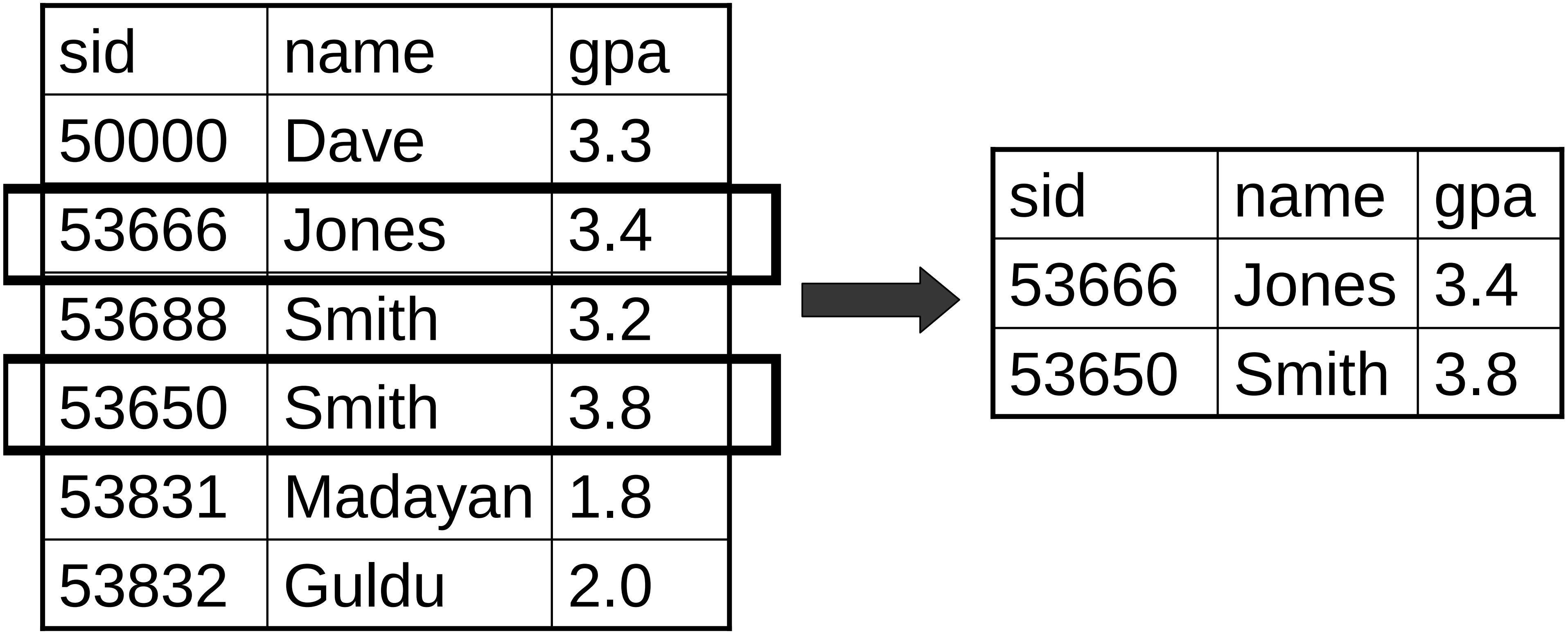
- Courses (cid, instructor, quarter, dept)
- Students (sid, name, gpa)
- Enrolled (cid, grade, studid)

Selection

Select students with gpa higher than 3.3 from S1*:*

$\sigma_{gpa>3.3}(S1)$

S1



sid	name	gpa
50000	Dave	3.3
53666	Jones	3.4
53688	Smith	3.2
53650	Smith	3.8
53831	Madayan	1.8
53832	Guldu	2.0

sid	name	gpa
53666	Jones	3.4
53650	Smith	3.8

Projection

Project name and gpa of all students in S1:

$\Pi_{\text{name, }\mathit{gpa}}(\text{S1})$

S1

Sid	name	gpa
50000	Dave	3.3
53666	Jones	3.4
53688	Smith	3.2
53650	Smith	3.8
53831	Madayan	1.8
53832	Guldu	2.0

name	gpa
Dave	3.3
Jones	3.4
Smith	3.2
Smith	3.8
Madayan	1.8
Guldu	2.0

Combine Selection and Projection

- Project name and gpa of students in S1 with gpa higher than 3.3:

$$\Pi_{\mathrm{name},\mathit{gpa}}(\sigma_{\mathit{gpa}>3.3}(\mathrm{S1}))$$

Sid	name	gpa
50000	Dave	3.3
53666	Jones	3.4
53688	Smith	3.2
53650	Smith	3.8
53831	Madayan	1.8
53832	Guldu	2.0

name	gpa
Jones	3.4
Smith	3.8

Set Operations

- Union (R U S)
  - All tuples in R or S (or both)
  - R and S must have same number of fields
  - Corresponding fields must have same domains
- Intersection (R ∩ S)
  - All tuples in both R and S
- Set difference (R – S)
  - Tuples in R and not S

Set Operations (continued)

- Cross product or Cartesian product (R x S)
  - All fields in R followed by all fields in S
  - One tuple (r,s) for each pair of tuples r ∈ R, s ∈ S

Example: Intersection

S1

sid	name	gpa
50000	Dave	3.3
53666	Jones	3.4
53688	Smith	3.2
53650	Smith	3.8
53831	Madayan	1.8
53832	Guldu	2.0

S2

sid	name	gpa
53666	Jones	3.4
53688	Smith	3.2
53700	Tom	3.5
53777	Jerry	2.8
53832	Guldu	2.0

$S1 \cap S2 =$

sid	name	gpa
53666	Jones	3.4
53688	Smith	3.2
53832	Guldu	2.0

Joins

- Combine information from two or more tables
- Example: students enrolled in courses:

$S1 \bowtie_{S1.sid=E.studid} E$

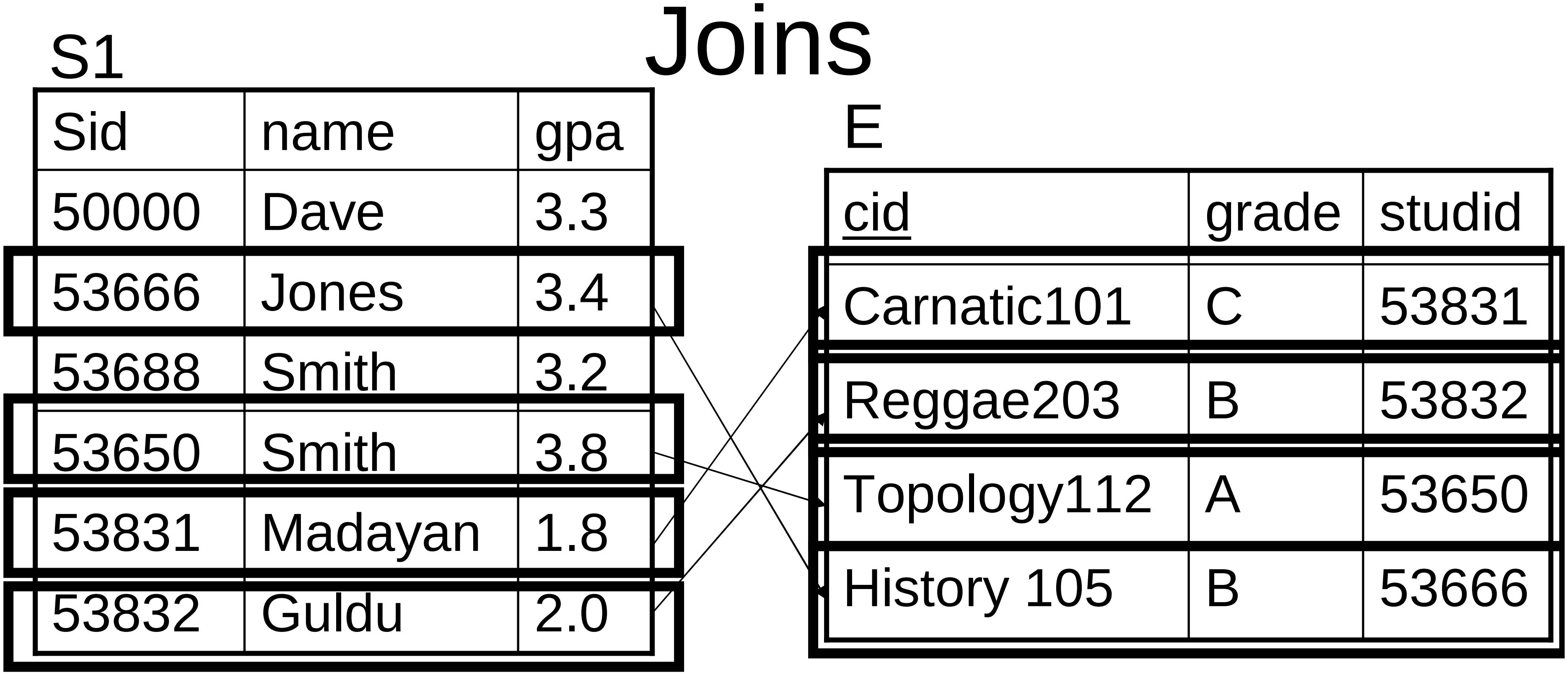
S1

Sid	name	gpa
50000	Dave	3.3
53666	Jones	3.4
53688	Smith	3.2
53650	Smith	3.8
53831	Madayan	1.8
53832	Guldu	2.0

E

cid	grade	studid
Carnatic101	C	53831
Reggae203	B	53832
Topology112	A	53650
History 105	B	53666

Joins



S1

Sid	name	gpa
50000	Dave	3.3
53666	Jones	3.4
53688	Smith	3.2
53650	Smith	3.8
53831	Madayan	1.8
53832	Guldu	2.0

E

cid	grade	studid
Carnatic101	C	53831
Reggae203	B	53832
Topology112	A	53650
History 105	B	53666

sid	name	gpa	cid	grade	studid
53666	Jones	3.4	History105	B	53666
53650	Smith	3.8	Topology112	A	53650
53831	Madayan	1.8	Carnatic101	C	53831
53832	Guldu	2.0	Reggae203	B	53832

Relational Algebra Summary

- Algebras are useful to manipulate data types (relations in this case)
- Set-oriented
- Brings some clarity to what needs to be done
- Opportunities for optimization
  - May have different expressions that do same thing
- We will see examples of algebras for other types of data in this course

Intro to SQL

- CREATE TABLE
  - Create a new table, e.g., students, courses, enrolled
- SELECT-FROM-WHERE
  - List all CS courses
- INSERT
  - Add a new student, course, or enroll a student in a course

Create Table

- CREATE TABLE Enrolled

```
(studid CHAR(20),
cid CHAR(20),
grade CHAR(20),
PRIMARY KEY (studid, cid),
 FOREIGN KEY (studid) references Students)
```

Select-From-Where query

- “Find all students who are under 18”

```
SELECT *
FROM    Students S
WHERE  S.age < 18
```

Queries across multiple tables (joins)

- “Print the student name and course ID where the student received an ‘A’ in the course”

```
SELECT S.name, E.cid
FROM    Students S, Enrolled E
WHERE  S.sid = E.studid AND E.grade = ‘A’
```

Other SQL features

- MIN, MAX, AVG
  - Find highest grade in fall database course
- COUNT, DISTINCT
  - How many students enrolled in CS courses in the fall?
- ORDER BY, GROUP BY
  - Rank students by their grade in fall database course

Views

- Virtual table defined on base tables defined by a query
  - Single or multiple tables
- Security – “hide” certain attributes from users
  - Show students in each course but hide their grades
- Ease of use – expression that is more intuitively obvious to user
- Views can be materialized to improve query performance

Views

- Suppose we often need names of students who got a ‘B’ in some course:

```
CREATE VIEW B_Students(name, sid, course)
   AS SELECT S.sname, S.sid, E.cid
   FROM Students S, Enrolled E
   WHERE S.sid=E.studid and E.grade = ‘B’
```

name	sid	course
Jones	53666	History105
Guldu	53832	Reggae203

Indexes

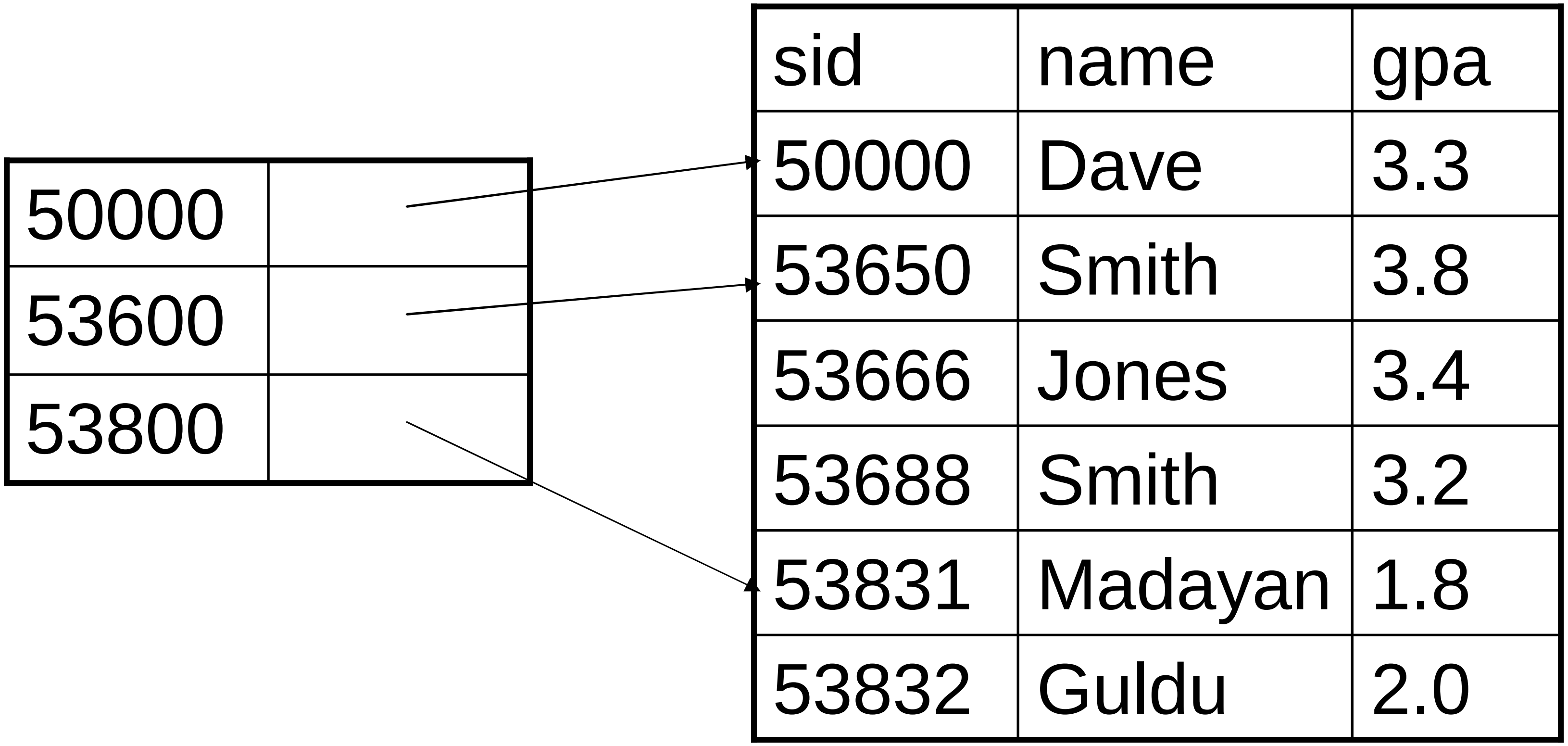
- Idea: speed up access to desired data
- “Find all students with gpa > 3.3
- May need to scan entire table
- Index consists of a set of *entries* pointing to locations of each *search key*

Types of Indexes

- Clustered vs. Unclustered
  - Clustered- ordering of data records same as ordering of data entries in the index
  - Unclustered- data records in different order from index
- Primary vs. Secondary
  - Primary – index on fields that include primary key
  - Secondary – other indexes

Example: Clustered Index

- Sorted by sid

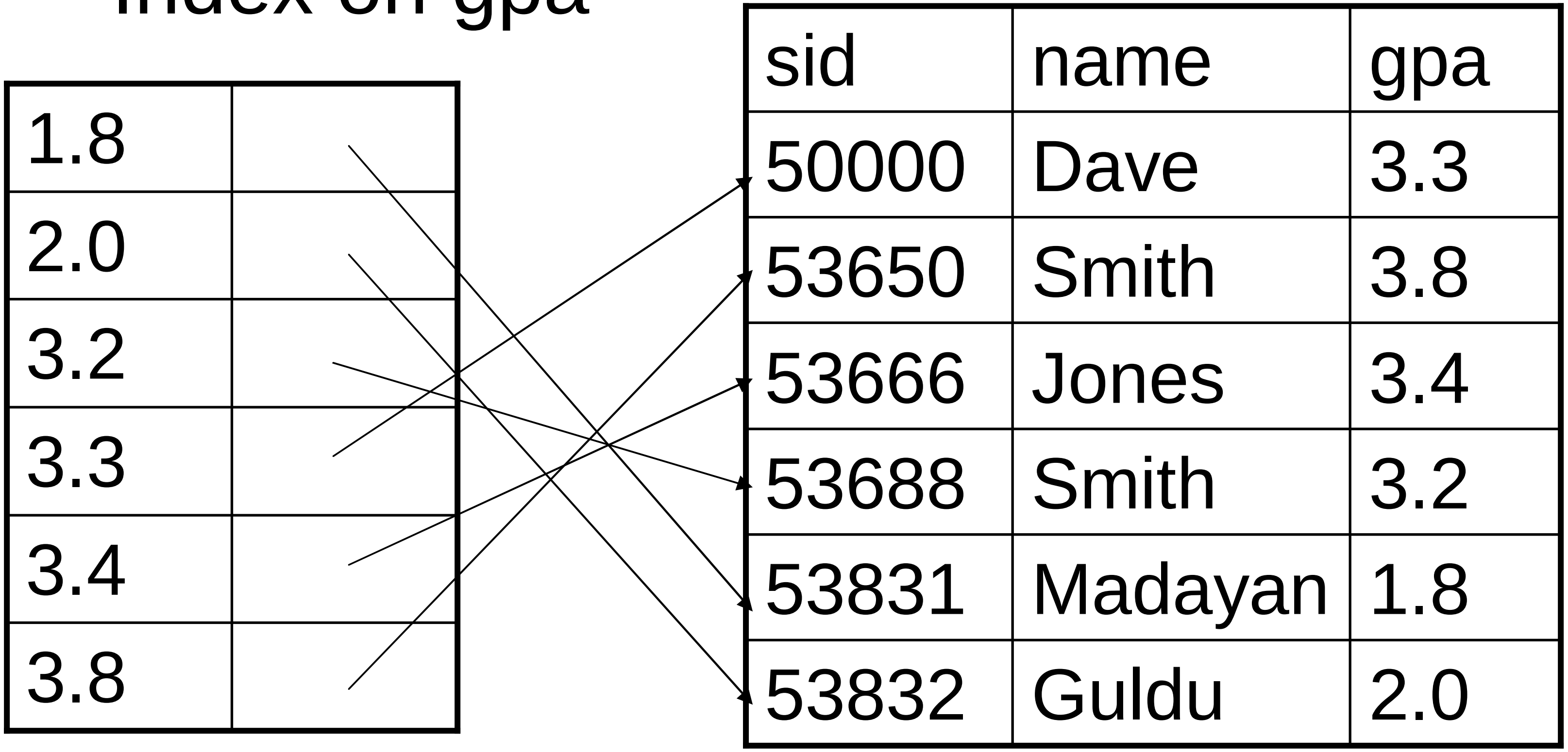


50000	
53600	
53800	

sid	name	gpa
50000	Dave	3.3
53650	Smith	3.8
53666	Jones	3.4
53688	Smith	3.2
53831	Madayan	1.8
53832	Guldu	2.0

Example: Unclustered Index

- Sorted by sid
- Index on gpa



1.8	
2.0	
3.2	
3.3	
3.4	
3.8	

sid	name	gpa
50000	Dave	3.3
53650	Smith	3.8
53666	Jones	3.4
53688	Smith	3.2
53831	Madayan	1.8
53832	Guldu	2.0

Comments on Indexes

- Indexes can significantly speed up query execution
- But inserts more costly
- May have high storage overhead
- Need to choose attributes to index wisely!
  - What queries are run most frequently?
  - What queries could benefit most from an index?
- Preview of things to come: SDSS

Summary: Why are RDBMS useful?

- Data independence – provides abstract view of the data, without details of storage
- Efficient data access – uses techniques to store and retrieve data efficiently
- Reduced application development time – many important functions already supported
- Centralized data administration
- Data Integrity and Security
- Concurrency control and recovery

So, why don’t scientists use them?

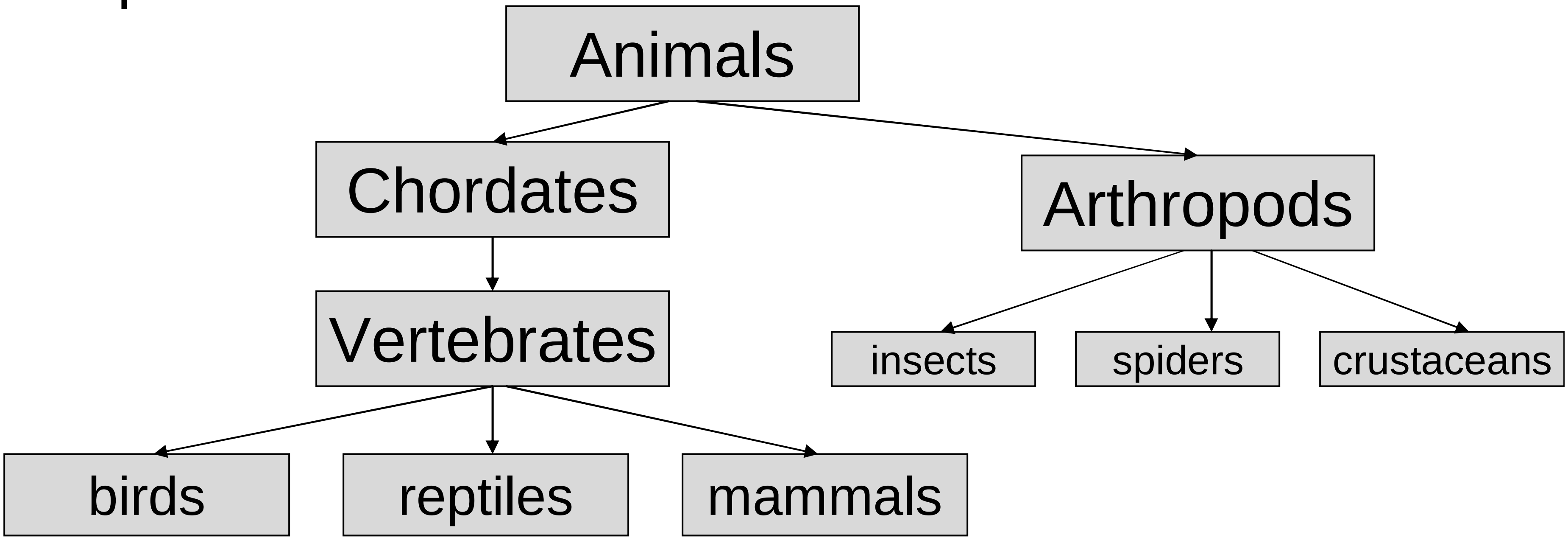
- “I tried to use databases in my project, but they were just too [slow | hard-to-use | expensive | complex] . So I use files”.
  - – Gray and Szalay, *Where Rubber Meets the Sky: Bridging the Gap Between Databases and Science*

Some other limitations of RDBMS

- Arrays
- Hierarchical data

Example: Taxonomy of Organisms

- Hierarchy of categories:
  - Kingdom - phylum – class – order – family – genus - species



  - How would you design a relational schema for this?